

卷积神经网络的直观解释

Posted on August 11, 2016 May 29, 2017 by ujjwalkarn

什么是卷积神经网络，为什么它们很重要？

卷积神经网络（ConvNets 或 CNN）是神经网络的一个类别，在图像识别和分类等领域已被证明非常有效。除了机器人和自驾车的视力供电之外，ConvNets 已经成功地识别了面部，物体和交通标志。



Figure 1: Source [1 (<http://cs.stanford.edu/people/karpathy/neuraltalk2/demo.html>)]

在上图 1 中，ConvNet 能够识别场景，系统能够建议相关字幕（“足球运动员正在踢足球”），而图 2 显示了 ConvNets 用于识别日常物体，人类和动物。最近，ConvNets 在几种自然语言处理任务（例如句子分类）中也是有效的。

2017/6/10

An Intuitive Explanation of Convolutional Neural Networks – the data science blog

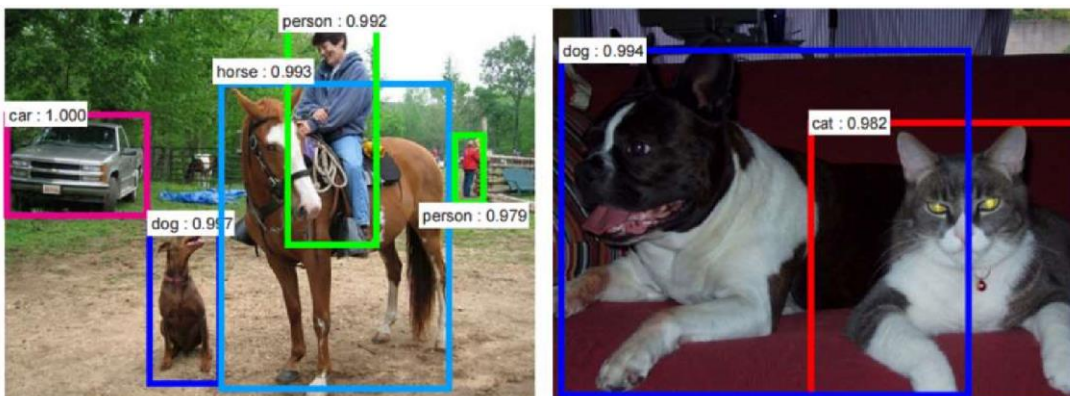


Figure 2: Source [2 (<https://arxiv.org/pdf/1506.01497v3.pdf>)]

因此，ConvNets 是当今大多数机器学习从业者的重要工具。然而，了解 ConvNets 和学习首次使用它们有时可能是一个令人恐惧的体验。这个博文的主要目的是开发一种对卷积神经网络如何在图像上工作的理解。

如果你是一般的神经网络的新人，我会推荐阅读这篇关于多层感知器的简短教程，以了解它们的工作原理，然后再继续。多层感知器在这篇文章中被称为“Fully Connected Layers”。

LeNet架构(1990s)

LeNet 是第一个卷积神经网络之一，有助于推动深度学习领域。自 1988 年以来，Yann LeCun 的这项开创性工作被评为 LeNet5 之后的许多成功迭代[3]。当时 LeNet 架构主要用于字符识别任务，如阅读邮政编码，数字等。

下面，我们将展开 LeNet 架构如何学习识别图像的直觉。近年来提出了几种新的架构，即 LeNet 的改进，但是他们都使用 LeNet 的主要概念，如果您对前者有一个清晰的了解，那么相对来说更容易理解。

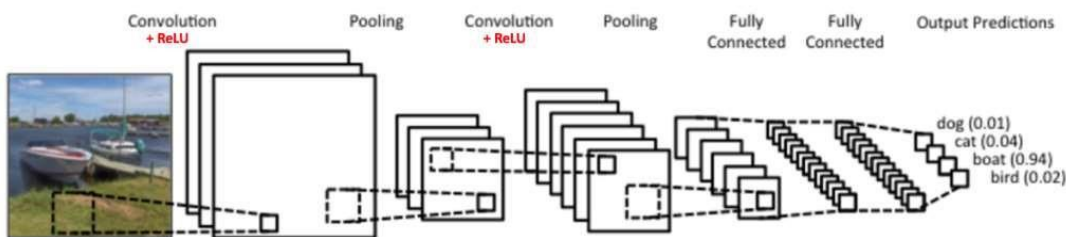


Figure 3: A simple ConvNet. Source [5 (<https://www.clarifai.com/technology/>)]

图 3 中的卷积神经网络在体系结构与原始 LeNet 类似，将输入图像分为四类：狗，猫，船或鸟（原始 LeNet 主要用于字符识别任务）。从上图可以看出，在接收到船只图像作为输入时，网络正确地分配了所有四个类别中船的最高概率（0.94）。输出层中所有概率的总和应为一（本文后面将会解释）。

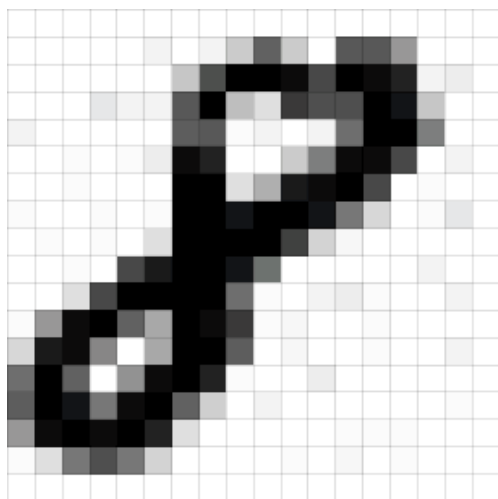
ConvNet 有四个主要操作，如图 3 所示：

1. Convolution
2. Non Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

这些操作是每个卷积神经网络的基本组成部分，因此了解这些工作是如何发展 ConvNets 的良好理解的重要一步。我们将尝试了解下面这些操作背后的细节。

图像是像素值的矩阵

本质上，每个图像可以表示为像素值的矩阵。



通道是用于指代图像的某个分量的常规术语。来自标准数码相机的图像将具有三个通道 - 红色，绿色和蓝色 - 您可以将其中的三个二维矩阵（每种颜色一个）堆叠起来，每个像素值的范围为 0 到 255。

另一方面，灰度图像只有一个通道。为了这个帖子的目的，我们只会考虑灰度图像，所以我们将有一个单一的 2d 矩阵代表一个图像。矩阵中每个像素的值范围为 0 到 255 - 零表示黑色，255 表示白色。

Convolution 卷积步骤

ConvNets 从“卷积”运算符中导出其名称。ConvNet 的主要目的是从输入图像中提取特征。卷积通过使用小平方的输入数据学习图像特征来保留像素之间的空间关系。我们不会在这里进入卷积的数学细节，但会尝试了解它如何在图像上工作。如上所述，每个图像都可以被认为是像素值的矩阵。考虑像素值仅为 0 和 1 的 5 x 5

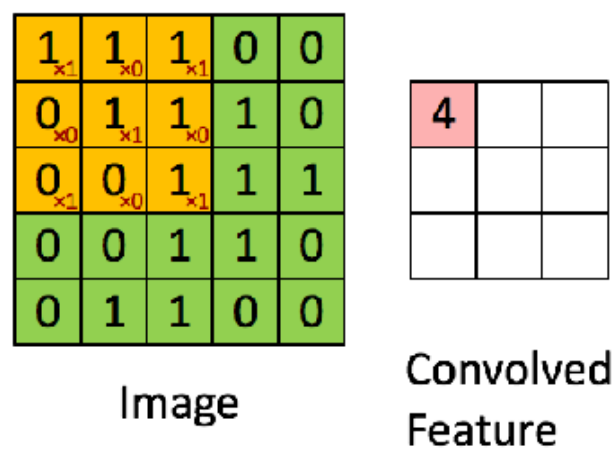
图像（请注意，对于灰度图像，像素值范围为 0 到 255，下面的绿色矩阵是像素值仅为 0 和 1 的特殊情况）：

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

另外，考虑另外 3×3 矩阵，如下所示：

1	0	1
0	1	0
1	0	1

然后，可以如下面的图 5 中的动画所示计算 5 x 5 图像和 3 x 3 矩阵的卷积：



花点时间了解上面的计算是如何进行的。我们将橙色矩阵滑过我们的原始图像（绿色）

1 像素（也称为“stride”），对于每个位置，我们计算元素乘法（在两个矩阵之间），并加上乘法输出以得到最终的整数 输出矩阵的单个元素（粉红色）。请注意，3×3 矩阵在每个步幅中只能看到输入图像的一部分。








在 CNN 术语中，3×3 矩阵称为“过滤器 filter”或“内核 kernel”或“特征检测器 feature detector”，通过将图像上的滤镜滑动并计算点积形成的矩阵称为“卷积特征 Convolved Feature”或“Activation Map “或” Feature Map “。重要的是注意，滤波器作为来自原始输入图像的特征检测器。

从上面的动画可以看出，滤波器矩阵的不同值将为相同的输入图像产生不同的特征图。例如，考虑以下输入图像：



在下表中，我们可以看到上述图像与不同滤镜的卷积效应。如图所示，我们可以通过在卷积运算之前更改滤波器矩阵的数值来执行边缘检测，锐化和模糊等操作[8]

– 这意味着不同的滤波器可以从图像中检测不同的特征，例如边缘， 曲线等。

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

了解卷积运算的另一个好方法是通过查看下面图 6 中的动画：



Figure 6: The Convolution Operation. Source [9
(http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/)]

滤镜（带有红色轮廓）在输入图像上滑动（卷积运算）以产生特征图。另一个滤镜（带有绿色轮廓）在同一图像上的卷积给出了不同的特征图，如图所示。重要的是要注意，卷积运算捕获原始图像中的本地依赖关系。还要注意这两个不同的滤镜如何从相同的原始图像中生成不同的特征图。请记住，上面的图像和上面的两个滤镜只是数字矩阵，如上所述。

实际上，CNN 在训练过程中自己学习这些过滤器的价值（尽管我们还需要在训练过程之前指定过滤器数量，过滤器尺寸，网络架构等参数）。我们拥有的滤镜数量越多，图像特征越多越好，网络越能识别图像中的图像。

特征图（卷积特征）的大小由在卷积步骤执行之前需要决定的三个参数[4]控制：

- **Depth:**深度对应于我们用于卷积运算的滤波器的数量。在图 7 所示的网络中，我们使用三个不同的滤波器对原始船只图像进行卷积，从而产生三个不同的特征图，如图所示。您可以将这三个特征图视为堆叠的 2d 矩阵，因此，特征图的“深度”将为 3。

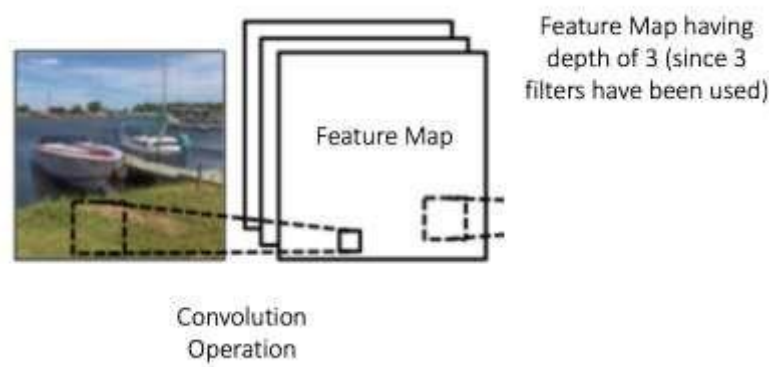


Figure 7

- **Stride:** Stride 是我们通过输入矩阵滑动我们的滤波器矩阵的像素数。当步幅为 1 时，我们一次移动一个像素的滤镜。当步幅为 2 时，当我们滑动它们时，滤镜会一次跳过 2 像素。更大的步幅将产生较小的特征图。
- **Zero-padding:**有时，在边框周围用零填充输入矩阵是方便的，所以我们可以将滤波器应用到我们输入图像矩阵的边界元素。零填充的一个很好的特点是它允许我们控制特征图的大小。添加零填充也称为宽卷积，不使用零填充将是一个窄卷积。这已经在[14]中得到了清楚的解释。

[14] (<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>).

引入非线性（ReLU）

在上面的图 3 中的每个卷积操作之后，已经使用了称为 ReLU 的附加操作。ReLU 代表整流线性单元，是非线性运算。其输出由下式给出：

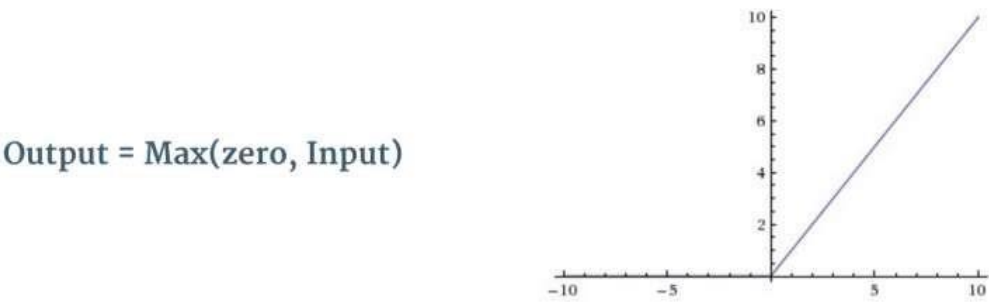


Figure 8: the ReLU operation

ReLU 是元素操作（每像素应用），并将特征图中的所有负像素值替换为零。ReLU 的目的是在 ConvNet 中引入非线性，因为我们希望 ConvNet 学习的大多数真实世界的数据将是非线性的（卷积是一个线性运算 - 元素矩阵乘法和加法，所以我们通过引入诸如 ReLU 的非线性函数来解释非线性）。

ReLU 操作可以从下面的图 9 清楚地看出。它显示了 ReLU 操作应用于上述图 6 中获得的特征图之一。此处的输出特征图也称为“整流”特征图。

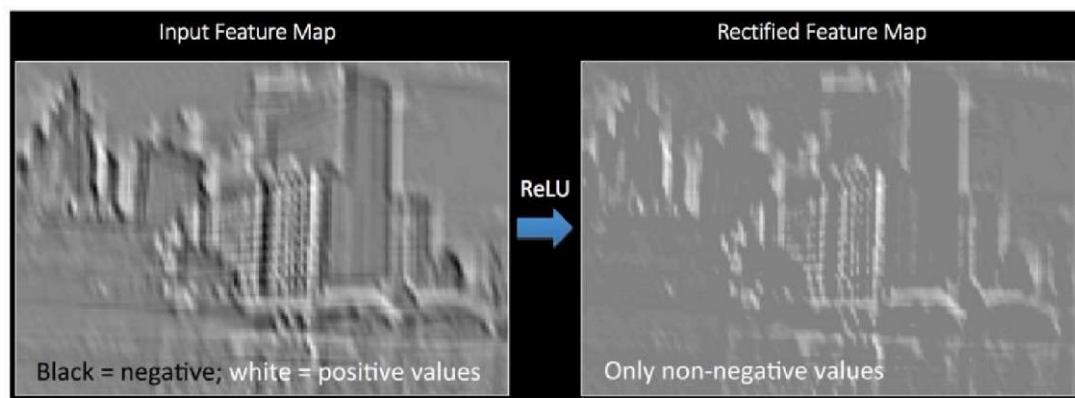


Figure 9: ReLU operation. Source [10
(http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf)]

还可以使用其他非线性函数（如 tanh 或 sigmoid）来代替 ReLU，但是在大多数情况下，ReLU 已被发现表现更好。

Pooling 步骤

Spatial Pooling 空间汇总（也称为次采样 subsampling 或下采样 downsampling）降低了每个特征图的维数，但保留了最重要的信息。空间汇总可以是不同的类型：最大，平均，总和等

在 Max Pooling 的情况下，我们定义一个空间邻域（例如，一个 2×2 窗口），并从该窗口内的整流特征图中获取最大的元素。而不是使用最大的元素，我们也可以取平均值（Average Pooling）或该窗口中所有元素的总和。实际上，Max Pooling 已经显示出更好的效果。

图 10 显示了使用 2×2 窗口的“整流功能”映射（通过卷积+ ReLU 操作后获得）上的 Max Pooling 操作的示例。

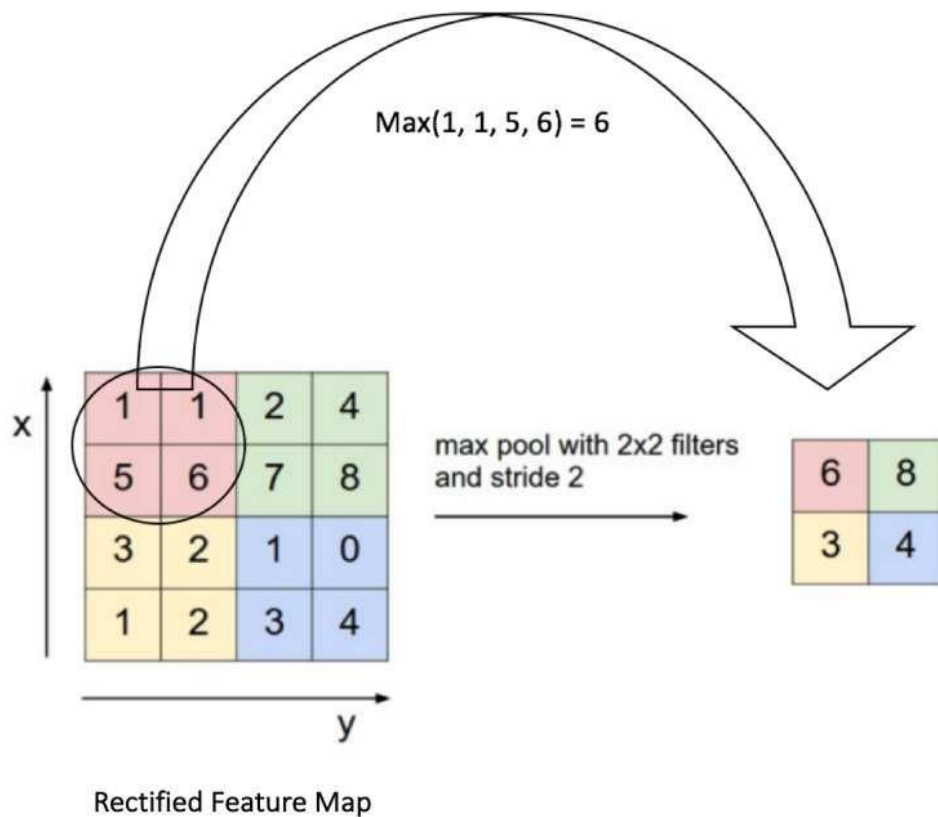


Figure 10: Max Pooling. Source [4 (<http://cs231n.github.io/convolutional-networks/>)]

我们将 2×2 窗口滑动 2 个单元格（也称为“stride”），并在每个区域中取最大值。如图 10 所示，这降低了我们的特征图的维数。

在图 11 所示的网络中，pooling 操作分别应用于每个特征图（注意，由于这一点，我们从三个输入图中得到三个输出图）。

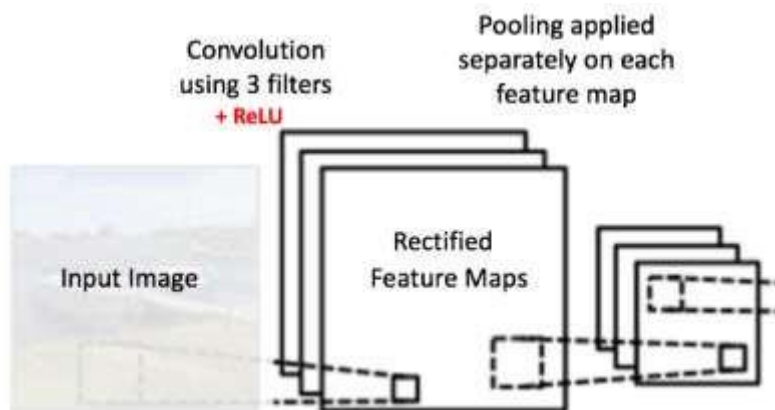


Figure 11: Pooling applied to Rectified Feature Maps

图 12 显示了在上图 9 中 ReLU 操作之后，我们收到的整流功能映射的汇总效果。

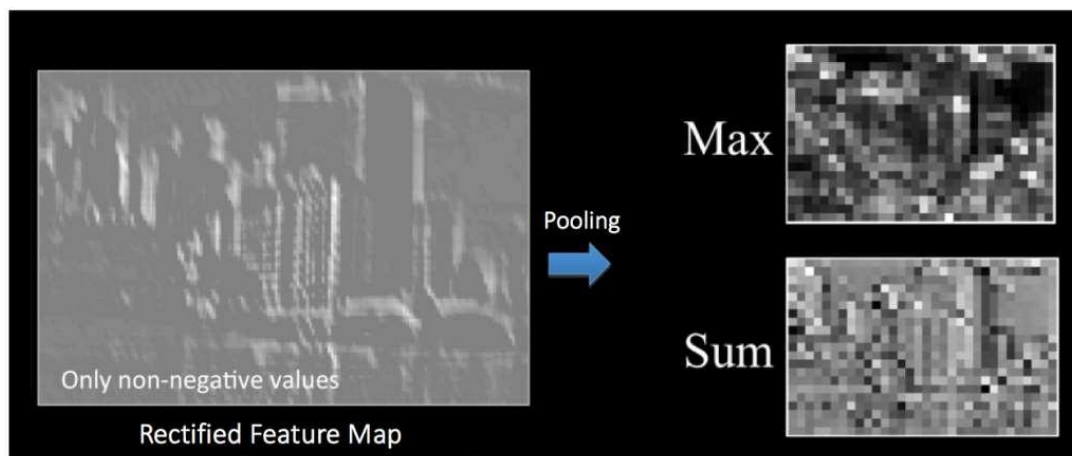


Figure 12: Pooling. Source [10 (http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf)]

Pooling 的功能是逐步减少输入表示的空间大小[4]。

- 使得输入表示（特征维度）更小更易于管理
- 减少网络中的参数和计算次数，从而控制过度拟合[4]
[4 (<http://cs231n.github.io/convolutional-networks/>)]
- 使得网络忽略输入图像中的小变换，失真和转换（输入中的小失真不会改变 Pooling 的输出 - 因为我们在本地邻域中取最大/平均值）。
- 帮助我们得到图片几何比例不变表示（确切的术语是“等量的”）。这是非常强大的，因为我们可以检测图像中的对象，无论它们位于何处（详见[18]和[19]）。
[18 (<https://github.com/rasbt/python-machine-learning-book/blob/master/faq/difference-deep-and-normal-learning.md>)] and [19 (<https://www.quora.com/How-is-a-convolutional-neural-network-able-to-learn-invariant-features>)] for details).

到目前为止的故事

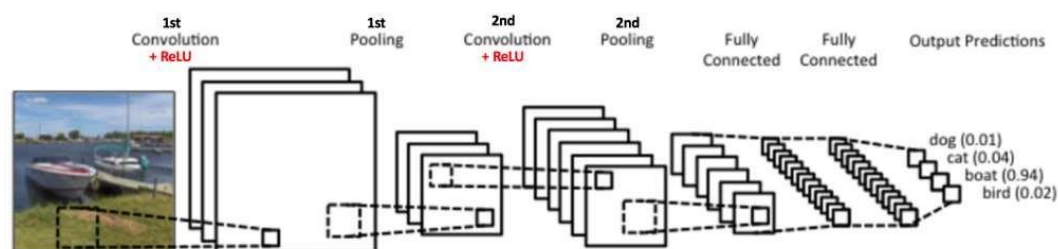


Figure 13

到目前为止，我们已经看到了卷积，ReLU 和 Pooling 如何工作。重要的是要了解这些层是任何 CNN 的基本构件。如图 13 所示，我们两组卷积，ReLU&Pooling 层 - 第二卷

卷层使用六个滤波器对第一个 Pooling Layer 的输出进行卷积，以产生总共六个特征图。然后将 ReLU 单独应用于所有这六个特征图。然后，我们分别对六个整流特征图中的每一个执行 Max Pooling 操作。这些层一起从图像中提取有用的特征，在我们的

网络中引入非线性，并减少特征维度，同时旨在使特征与缩放和平移具有一些等同性 [18]。

[18] (<https://github.com/rasbt/python-machine-learning-book/blob/master/faq/difference-deep-and-normal-learning.md>).

第二个 Pooling 层的输出作为全连接层 Fully Connected Layer 的输入，我们将在下一节讨论。

Fully Connected Layer

全连接层是传统的多层感知机，它在输出层中使用 softmax 激活函数（也可以使用其他分类器，如 SVM，但是在这篇文章中会坚持 softmax）。术语“全连接”意味着上一层中的每个神经元连接到下一层的每个神经元。如果您不熟悉多层感知机，我建议您阅读这篇文章。

(<https://uijwalkarn.me/2016/08/09/quick-intro-neural-networks/>)

卷积层和 pooling 层的输出代表输入图像的高级特征。全连接层的目的是使用这些功能将输入图像分类为基于训练数据集的各种类别。例如，我们设定执行的图像分类任务具有四个可能的输出，如下面的图 14 所示（请注意，图 14 未显示完全连接的图层中的节点之间的连接）

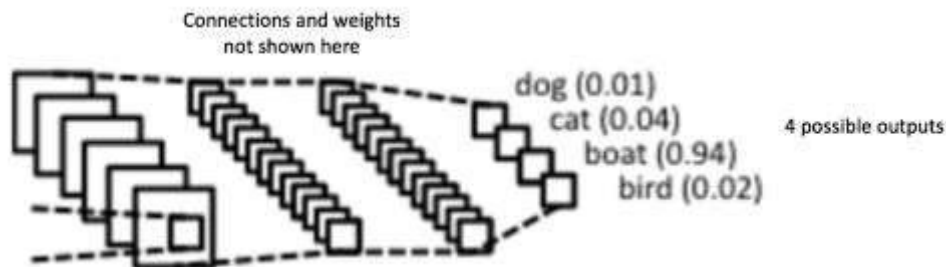


Figure 14: 全连接层 – 每个节点连接到相邻层中的每个其他节点

除了分类之外，添加全连接层也是学习这些特征的非线性组合的（通常）廉价方法。来自卷积层和 pooling 层的大部分特征对于分类任务可能是好的，但是这些特征的组合可能更好 [11]。

[11] (<https://stats.stackexchange.com/questions/182102/what-do-the-fully-connected-layers-do-in-cnns/182122#182122> .)]

来自全连接层的输出概率的总和为 1. 通过使用 Softmax 作为完全连接层的输出层中的激活功能来确保。Softmax 函数采用任意实值分数的向量，并将其压缩为零和一之间的值向量，一个总和为 1。

Softmax (<http://cs231n.github.io/linear-classify/#softmax>)

把它放在一起 - 使用反向传播 Backpropagation 训练

如上所述，卷积+Pooling 层作为输入图像的特征提取器，而全连接层用作分类器。

注意，在下面的图 15 中，由于输入图像是船，所以船级别的目标概率为 1，对于其他三类，目标概率为 0，即

- Input Image = Boat
- Target Vector = [0, 0, 1, 0]

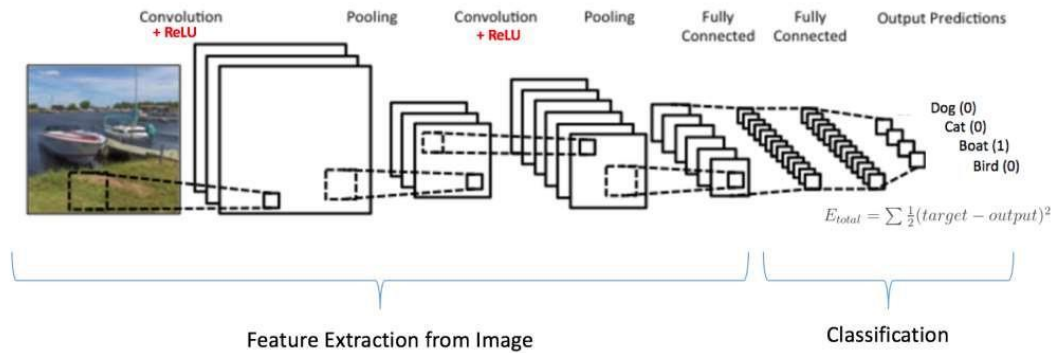


Figure 15: Training the ConvNet

卷积网络的整体训练过程可概述如下：

- 步骤 1：我们用随机值初始化所有过滤器和参数/权重
- 步骤 2：网络采用训练图像作为输入，通过正向传播步骤（卷积，ReLU 和 pooling 操作以及完全连接层中的正向传播），并找到每个类的输出概率。
 - 让我们说上面的船图像的输出概率是 [0.2, 0.4, 0.1, 0.3]
 - 由于权重被随机分配给第一训练样本，输出概率也是随机的。
- 步骤 3：计算输出层的总误差（所有 4 个类的总和）总误差 $= \sum \frac{1}{2} (\text{目标概率} - \text{输出概率})^2$
- 步骤 4：使用反向传播来计算相对于网络中所有权重的错误梯度，并使用梯度下降来更新所有过滤器值/权重和参数值以最小化输出错误。
 - 权重与其对总错误的贡献成比例地调整。
 - 当再次输入相同的图像时，输出概率现在可以是 [0.1, 0.1, 0.7, 0.1]，这更接近目标矢量 [0, 0, 1, 0]
 - 这意味着网络已经学会了通过调整其权重/滤波器来正确地分类该特定图像，使得输出误差减小。
- 滤波器数量，滤波器大小，网络架构等参数在步骤 1 之前都被设置好，在训练过程中不改变，只有滤波器矩阵和连接权重的值得到更新。
- 步骤 5：对训练集中的所有图像重复步骤 2-4。

上述步骤训练 ConvNet - 这实质上意味着 ConvNet 的所有权重和参数现在已经被优化，以正确地对训练集中的图像进行分类。

当 ConvNet 中输入新的（不可见的）图像时，网络将通过正向传播步骤并输出每个类的概率（对于新图像，输出概率是使用已优化的权重计算得到的，以正确分类所有以前的训练示例）。如果我们的训练集足够大，网络将（希望）推广到新的图像，并将其分类为正确的类别。

注 1：上述步骤过于简单，避免了数学细节，为训练过程提供直观的理解。参见 [4] 和 [12] 的数学公式和透彻的理解。[4 (<http://cs231n.github.io/convolutional-networks/>)] [12 (<http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/>)]

注 2：在上面的例子中，我们使用了两组交替的卷积和 Pooling 层。但请注意，这些操作可以在单个 ConvNet 中重复任意次数。事实上，一些表现最好的 ConvNets 今天拥有数十个卷积和 Pooling 层！此外，在每个卷积层之后没有必要一定有 Pooling 层。从下面的图

16 可以看出，在进行 Pooling 操作之前，我们可以连续进行多次卷积+重新执行操作。还要注意 ConvNet 的每一层如下图 16 所示。

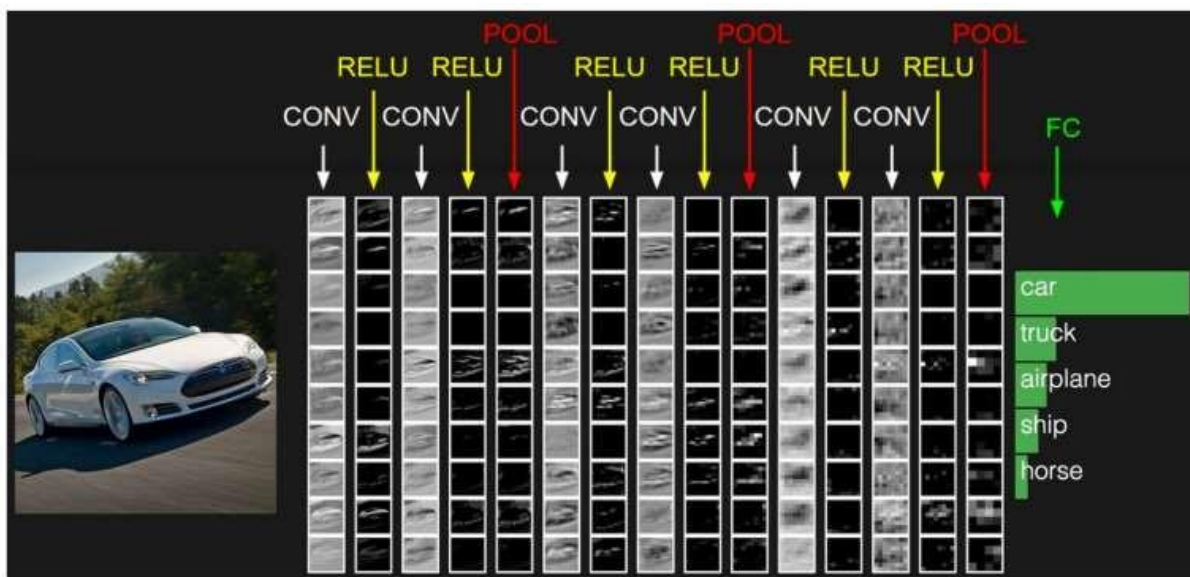


Figure 16: Source [4 (<http://cs231n.github.io/convolutional-networks/>)]

可视化卷积神经网络

一般来说，我们拥有的卷积步骤越多，我们的网络就越能复杂地识别。例如，在图像分类中，ConvNet 可以学习检测第一层中原始像素的边缘，然后使用边缘来检测第二层中的简单形状，然后使用这些形状来组成更高级的特征，如面部形状在较高层 [14]。

[14 (<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>)]

并且该图包括在此仅仅是为了演示这个想法（这仅仅是一个例子：现实生活中的卷积滤波器可以检测到对人没有意义的对象）。这在下面的图 17 中被证明 - 这些特征是使用 Convolutional Deep Belief 学习的，并且该图包括在此仅仅是为了演示这个想法（这仅仅是一个例子：现实生活中的卷积滤波器可以检测到对人没有意义的对象）。

(<http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf>)

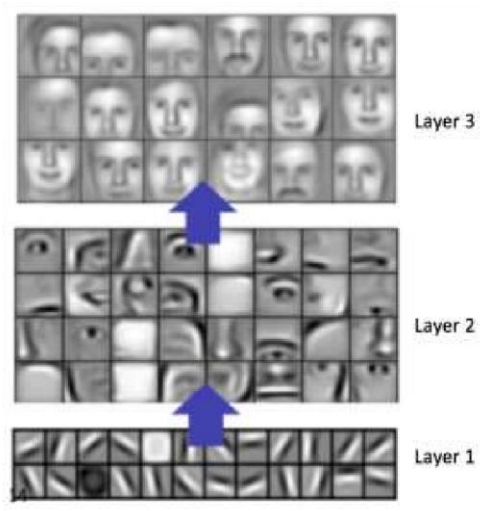


Figure 17: Learned features from a Convolutional Deep Belief Network. Source [21] (<http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf>)

Adam Harley 创建了一个由 MNIST 数据库手写数字训练的卷积神经网络的惊人可视化功能[13]。我强烈推荐玩弄它来了解 CNN 如何工作的细节。

我们将在下面看到如何为输入“8”工作的网络。请注意，图 18 中的可视化不会单独显示 ReLU 操作。

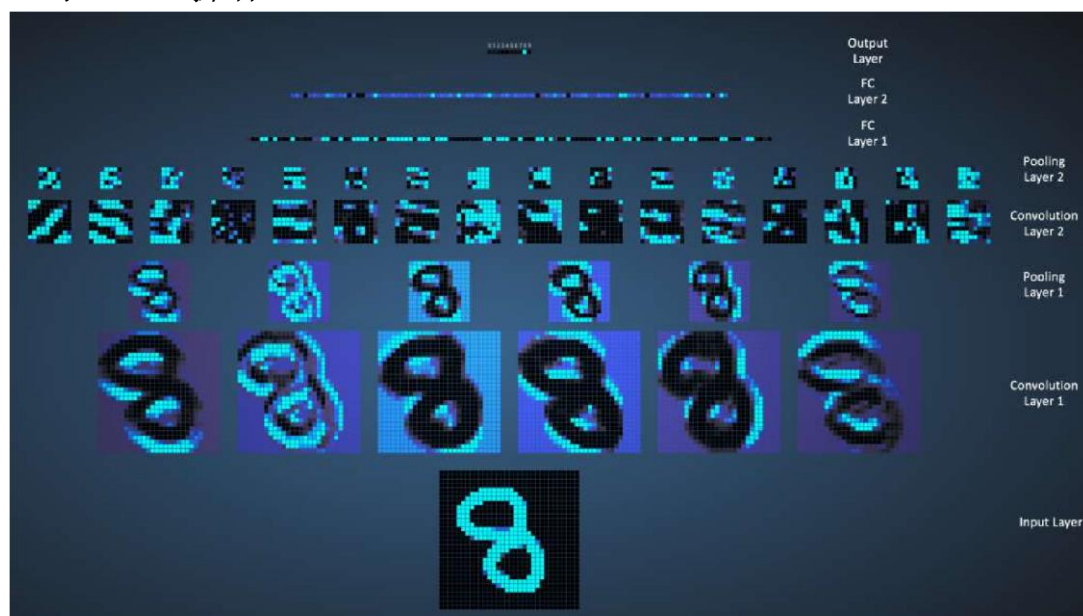


Figure 18: Visualizing a ConvNet trained on handwritten digits. Source [13] (<http://scs.ryerson.ca/~aharley/vis/conv/flat.html>)

输入图像包含 1024 像素（ 32×32 图像），并且通过六个唯一的 5×5 （步幅 1）滤波器与输入图像的卷积形成第一卷积层（卷积层 1）。如图所示，使用六个不同的滤波器产生深度为 6 的特征图。

卷积层 1 之后是 Pooling 层 1，它在卷积层 1 中的六个特征图中分别进行 2×2 max pooling（带有步幅 2）。您可以将鼠标指针移动到 pooling 层中的任何像素上，并观察 2×2 网格在前一个卷积层中形成（如图 19 所示）。您将注意到， 2×2 网格中具有最大值（最亮的像素）的像素将会进入 pooling 层。



Figure 19: Visualizing the Pooling Operation. Source [13]
(<http://scs.ryerson.ca/~aharley/vis/conv/flat.html>)

Pooling 层 1 之后是执行卷积运算的十六个 5×5 （步幅 1）卷积滤波器。其次是 Pooling

Layer2，它的 max pooling 为 2×2 （步幅 2）。这两层使用与上述相同的概念。然后我们有三个全连接（FC）层。有：

- 第一个 FC 层中有 120 个神经元第二个 FC 层的 100 个
- 神经元
- 10 个神经元在第三个 FC 层对应 10 位数 - 也称为输出层

请注意，如图 20 所示，输出层中的 10 个节点中的每一个都连接到第 2 个完全连接层中的所有 100 个节点（因此名称为“全连接”）。

此外，请注意输出层中唯一的亮点对应于“8” - 这意味着网络正确地对我们的手写数字进行了分类（亮点表示其输出较高，即 8 在所有其他数字中的概率最高）。

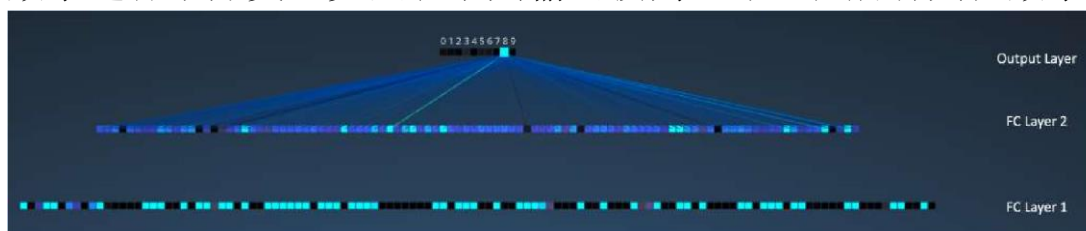


Figure 20: Visualizing the Fully Connected Layers. Source [13]
(<http://scs.ryerson.ca/~aharley/vis/conv/flat.html>)

相同可视化的3d版本可用 [here](http://scs.ryerson.ca/~aharley/vis/conv/) (<http://scs.ryerson.ca/~aharley/vis/conv/>).

其他 ConvNet 架构

卷积神经网络始于 20 世纪 90 年代初。我们讨论了 LeNet，这是最早的卷积神经网络之一。其他一些有影响力的架构如下 [3] [4] 所列。

- [3] (<https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba>)
- [4] (<http://cs231n.github.io/convolutional-networks/>).

- **LeNet (1990s):** 已在本文中介绍。

- **1990s to 2012:** 在 20 世纪 90 年代后期到 2010 年初，卷积神经网络正在孵化。随着越来越多的数据和计算能力的提高，卷积神经网络可以解决的任务变得越来越有趣。

- **AlexNet (2012)**
<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
 2012 年, Alex Krizhevsky (和其他人) 发布了 AlexNet, 这是 LeNet 的一个更深入和更广泛的版本, 并在 2012 年获得了巨大的困难的 ImageNet 大型视觉识别挑战 (ILSVRC)。这是一个重大的突破。以前的方法和 CNN 的当前广泛应用可以归功于这项工作。
- **ZF Net (2013)** – ILSVRC 2013 获奖者是 Matthew Zeiler 和 Rob Fergus 的卷积网络。它被称为 ZFNet (Zeiler & Fergus Net 的缩写)。这是通过调整架构超参数来改进 AlexNet
[ZFNet \(http://arxiv.org/abs/1311.2901\)](http://arxiv.org/abs/1311.2901)
- **GoogLeNet (2014)** –
 ILSVRC 2014 获奖者是 Szegedy 等人的卷积网络。来自 Google。其主要贡献是开发一个初始模块, 大大减少了网络中的参数数量 (4M, 与 AlexNet 的 60M 相比)。
<http://arxiv.org/abs/1409.4842>
- **VGGNet (2014)** – ILSVRC 2014 年的亚军是被称为 VGGNet 的网络。它的主要贡献在于显示网络的深度 (层数) 是良好性能的关键组成部分。
[VGGNet \(http://www.robots.ox.ac.uk/~vgg/research/very_deep/\)](http://www.robots.ox.ac.uk/~vgg/research/very_deep/).
- **ResNets (2015)** Kaiming He (和其他人) 开发的剩余网络是 ILSVRC 2015 的获胜者。ResNets 目前是迄今为止最先进的卷积神经网络模型, 是实际使用 ConvNets (截至 2016 年 5 月) 的默认选择。
[Residual Network \(http://arxiv.org/abs/1512.03385\)](http://arxiv.org/abs/1512.03385)
- **DenseNet (August 2016)** – 最近由高黄 (等) 发表, 密集卷积网络每层都以前瞻的方式直接连接到每一层。DenseNet 已经被证明可以在五个竞争激烈的对象识别基准测试任务上获得先前最先进的架构的显着改进。在这里查看 Torch 实现。
<http://arxiv.org/abs/1608.06993>
<https://github.com/liuzhuang13/DenseNet>.